

APPLICATION FOR LETTERS PATENT OF THE UNITED STATES

Michael Soemo  
338 N. Craig Place  
Lombard, IL 60148

Mark Gagner  
30W515 Diversey Parkway  
West Chicago, IL 60185

John Stewart  
5195 Morningview Drive  
Hoffman Estates, IL 60192

Phil Pollock  
926 Donnelly Place  
McHenry, IL 60050

A PARTIALLY EMBEDDED DATABASE AND AN  
EMBEDDED DATABASE MANAGER FOR A CONTROL SYSTEM

TO WHOM IT MAY CONCERN, THE FOLLOWING IS A  
SPECIFICATION OF THE AFORESAID INVENTION

1           The present invention generally relates to a data storage system  
2   for a control system, and more particularly to a database that is partitioned  
3   between static and dynamic memory and a database manager that is stored  
4   in static memory.

Control systems are becoming increasingly more computerized. As a result, many of today's controllers include processors for processing control system data. However, processing power is only useful if adequate memory is available to support the processor when performing tasks.

In addition, the control system design process often involves designing the hardware and software separately. However, the hardware for a controller is often completed before the software required to operate the controller. As a result, the amount of memory required for the design is

1 usually estimated in advance. However, the amount of memory required to  
2 support the final version of the software often exceeds the memory  
3 estimations due to the number of advanced features being supplied by today's  
4 controllers. Moreover, designing the hardware at the same time as the  
5 software will not necessarily eliminate the problem because, even after  
6 system installation, the software associated with a controller often continues  
7 to evolve to meet customer demands for additional features. As a result,  
8 control system designers are frequently forced to eliminate features or  
9 otherwise reduce the memory requirements of the software that supplies the  
10 features.

11 Thus, there is a need in the art for a device that overcomes one  
12 or more of the foregoing problems.

13

#### 14 BRIEF DESCRIPTION OF THE DRAWINGS

15 FIGURE 1 is a control system having a set of network devices,  
16 including an application node, coupled to a communication network according  
17 to one aspect of the invention;

18 FIG. 2 is a block diagram of the application node of FIG. 1,  
19 having a database and a database manager, and a workstation that is also  
20 coupled to the communication network of FIG. 1 according to another aspect  
21 of the invention;

22 FIG. 3 is a block diagram of a static memory device and two  
23 dynamic memory devices disposed in the application node of FIGs. 1 and 2;

24 FIG. 4 is an illustration of the data records and data fields  
25 contained in the database of FIG. 2;

26 FIG. 5 is a flow chart representing a method for creating and  
27 installing the database and database manager of FIG. 2 according to another  
28 aspect of the invention;

29 FIG. 6 is a flow chart representing a method for initializing a  
30 database from table files to produce a database cache and record files.

31 FIG. 7 is a flow chart representing a method for compressing a  
32 database according to still another aspect of the invention; and,

1           FIG. 8 is a flow chart representing a method for modifying a set  
2 of static data elements stored in the database according to a still further  
3 aspect of the invention.

4           FIG. 9 is a schematic of a control system similar in most  
5 respects to that illustrated by FIG. 1, except that an external communications  
6 network is further present.

7

## 8                           SUMMARY OF THE INVENTION

9           The present invention is directed to a partially embedded  
10 database configured as a set of files stored in a static memory device and in a  
11 dynamic memory device. A catalog defines the structure of the database and  
12 identifies the data elements that are stored in the static memory and the data  
13 elements that are stored in the dynamic memory. An embedded database  
14 manager uses the catalog to create and maintain the database and further  
15 uses a file system to access the database files.

16           A method for modifying the static data includes the steps of  
17 copying the static data into a cache implemented using random access  
18 memory, modifying the data stored in the cache and then writing the modified  
19 data into the static memory. A method for compressing the database to  
20 conserve memory involves collectively storing Boolean data elements.

21

## 22                           DETAILED DESCRIPTION

23           Referring now to the drawings wherein like reference numerals  
24 refer to similar or identical parts throughout the several views, and more  
25 specifically to FIG. 1 thereof, a control network 10 for providing, for example,  
26 building control includes a communication network 12 to support  
27 communication between a set of network control devices including an  
28 application specific controller 14, a programmable equipment controller 16, an  
29 application node 18, an operator workstation 20, and an engineering and  
30 commissioning tool 22. The network control devices may further include a set  
31 of interfaces by which an operator may monitor/control the system including a

1 portable operator interface 24, a panel mount interface 26, and a low end  
2 human machine interface 28 having a small display and limited features.

3           The application specific controller 14 is configured to control a  
4 local mechanical and/or electronic device (not shown) associated with a  
5 specific application such as, for example, valve or damper actuation. In  
6 contrast, the programmable equipment controller 16 is configurable to control  
7 a local mechanical and/or electronic device (not shown) associated with any  
8 desired type of application. The application node 18 provides services to the  
9 other network devices such as scheduling, data logging, paging, printing,  
10 alarm management and routing and protocol conversion. The operator  
11 workstation 20 automatically uploads and downloads network image data and  
12 system data and includes a user interface by which users may access control  
13 system information. The workstation 20 may be adapted to provide graphics,  
14 exception reporting, diagnostics, report generation, display, printing and dial  
15 out.

16           System engineering and commissioning is performed via the  
17 engineering and commissioning tool 22 which may also be used to graphically  
18 program the programmable equipment controller 16. In addition, the  
19 engineering and commissioning tool 22 may be used to compile data,  
20 download configuration data, perform diagnostics, generate and display  
21 reports and upload/download system data.

22           Referring now to FIGs. 2 and 3, the application node 18 includes  
23 a volatile random access memory (VRAM) device 30, a non-volatile random  
24 access memory device (NVRAM) device 31 and a static memory 32A and  
25 32B. The VRAM 30 is erased during each power cycle but will retain stored  
26 values during a reset operation and may be implemented using, for example,  
27 a Toshiba TC554001 memory device. In contrast, the NVRAM 31 retains  
28 stored values via power supplied via a battery (not shown) or similar auxiliary  
29 power supply even when the application node 18 loses its main power supply  
30 (not shown). As a result, the values stored in the NVRAM 31 are retained  
31 until actively erased. The application node 18 also includes a static memory  
32 device 32, implemented using, for example, a flash memory device, that

1 retains stored values in a semi-permanent fashion. The static memory device  
2 32 is partitioned into a flash program memory 32A reserved for software  
3 applications 34, including a database manager program 36, and a flash data  
4 memory 32B reserved for static, i.e., non-changing, data. In addition, the  
5 application node 18 includes a neuron network processor 40 to enable  
6 communication on the network 12 and further includes a microprocessor 42,  
7 such as a Motorola 68302 microprocessor, for executing the applications 34  
8 stored in the application node 18.

9           Like the application node 18, the workstation 20 also includes a  
10 random access memory (RAM) device 25, a hard disk memory device 27, a  
11 neuron network processor 29 and a microprocessor 31. In addition, the  
12 workstation 20 includes a database generation tool 43 for generating a  
13 database 38 and a database interface program 44 that may be used by an  
14 operator to access the database 38 disposed in the application node 18 via  
15 the network 12.

16           Referring still to FIG. 3, the database 38 is configured as a set of  
17 files 38A, 38B, 38C and 38D. Dynamic record files 38C and 38D are each  
18 stored on a memory device 31, with the two files occupying a contiguous  
19 area of the memory device 31. Each database table 38A and 38B includes a  
20 header of data referred to as a catalog 49 that defines the structure of the  
21 database file 38A and 38B and the type of data to be stored in each file 38A,  
22 38B, 38C, and 38D. Specifically, static table data and the dynamic table  
23 catalog, which includes data elements that are never or rarely expected to  
24 change, are stored in the static data files 38A and 38B located in the flash  
25 data memory 32B. Dynamic record data, which includes data elements that  
26 are expected to change frequently, are stored in two different types of files  
27 38C and 38D on NVRAM 31 depending on whether the data is categorized as  
28 persistent dynamic data or non-persistent dynamic data. The persistent  
29 dynamic data comprises the data elements that, although expected to change  
30 eventually, are also expected to remain in memory when power is removed  
31 from the application node 18. As a result, the persistent dynamic data are  
32 stored in the persistent dynamic data file 38D in the NVRAM 31. In contrast,

1 the non-persistent dynamic data are not expected to remain in memory when  
2 the power has been removed from the application node 18 and, therefore, the  
3 non-persistent dynamic data are stored in a non-persistent dynamic data file  
4 38C in the NVRAM 31. By way of example, the persistent dynamic data may  
5 comprise data such as accumulated totalization values for a process control  
6 device that must persist across a loss of power, and the non-persistent  
7 dynamic data may comprise data such as the current alarm state information  
8 for a process control device or the position of a valve or actuator.

9 Referring now to FIGs. 2, 3 and 4, the database files 38A, 38B,  
10 38C, and 38D contain process control information about the control system  
11 10. For example, the database files 38A, 38B, 38C, and 38D combined  
12 identify all of the process control devices coupled to the network 12 and  
13 provide information regarding the operation of each device. More specifically,  
14 the database files 38A, 38B, 38C, and 38D comprise a set of data records 46,  
15 each comprising a set of data fields 48. The catalog 49 indicates the number  
16 of data fields 48 in the database files 38A, 38B, 38C, 38D and the order in  
17 which the data fields 48 are stored in each of the database files 38A, 38B,  
18 38C, 38D. For example, a first data field 48a may be designated to store an  
19 identification number of a network device and a set of data fields 48b-48f  
20 following the first data field 48a may contain additional data related to the  
21 network device. The catalog 49 also indicates the characteristics of the data  
22 to be stored in each of the data fields 48. For example, the data fields 48 may  
23 be designated to contain character strings, integers, and/or numbers in a  
24 floating-point notation or Boolean elements. As will be appreciated by one  
25 having ordinary skill in the art, catalogs are conventionally used to define a  
26 database structure and may be created in any number of formats and may  
27 contain any desired information necessary to define the structure of the  
28 associated database.

29 The database manager 36 is a software program stored in the  
30 flash program memory 32A of the application node 18 that may be used to  
31 access data contained in the database 38. In particular, the database  
32 manager 36, operating in response to commands from the database interface

1 program 44, allows a user to retrieve the data contained in the database 38,  
2 to sort the data, modify the data and add or delete the data. Likewise, the  
3 application programs 34 stored in the application node 18 use the database  
4 manager 36 to access and utilize the database 38. The database manager  
5 36 may be configured to enable any number of advanced sorting and report  
6 generating features.

7           The database manager 36 accesses the database files 38A,  
8 38B, and 38C using a file system 37 that is implemented via software and that  
9 is stored in the flash program memory 32B of the application node 18.  
10 Information contained in database table files 38A and 38B direct the database  
11 manager to create cache areas 35 in VRAM and record files 38C and 38D in  
12 NVRAM. The file system 37 allows the database manager 36 to open and  
13 access each of the database files 38A, 38B, 38C, 38D and further allows the  
14 database manager 36 to directly access the memory contents of each  
15 database record file 38A, 38B, 38C, 38D as structured memory. More  
16 particularly, because the database record files 38C and 38D are stored as a  
17 set of contiguous files, the data stored in each file 38C and 38D remain  
18 unfragmented and arranged in a set order. As a result, the data can be  
19 accessed using any of a number of rapid addressing methods that generally  
20 involve using a memory pointer.

21           As will be appreciated by one having ordinary skill in the art, file  
22 systems are software tools that are conventionally used to open and  
23 manipulate computer files. However, file systems are typically designed to  
24 access a file by performing a series of steps, such as, for example, opening  
25 the file, seeking a position within a file, reading a segment of data from the  
26 file. Thus a series of steps are required before the desired data is actually  
27 obtained, causing these file access methods to be slow compared to methods  
28 that use direct memory pointers. Accordingly, the file system 37 provides  
29 access to the database file 38 using direct memory pointers to achieve  
30 timesavings.

31           In order to take advantage of this specialized functionality,  
32 database record files 38C and 38D are created as contiguous files. Normal



1 files hold their data as a linked list of many smaller blocks of data.  
2 Contiguous files, on the other hand, hold their data as a single large block of  
3 data in memory. The file system 37 provides a novel interface that returns the  
4 address of the single contiguous data block containing files 38C and 38D to  
5 the database manager 36. The database manager 36 is then free to access  
6 the contiguous data block as a structured memory without using the file  
7 system 37 as an intermediary. This novel functionality allows the present  
8 invention to accomplish significant timesavings over many methods of the  
9 prior art.

10           Because the database is configured as a set of files, any  
11 application stored on any other network device such as, for example, the  
12 workstation node 20, may directly communicate with the file system 37 and  
13 thereby access any of the database files 38A, 38B, 38C, and 38D. Since the  
14 files are preferably openly accessible using the standard file transfer protocol  
15 (FTP) additional communication strategies and communication protocols need  
16 not be developed to enable communication between an application on  
17 another network device and the file system operating in the application node.  
18 In addition, remote file system access features supported by the FTP protocol  
19 provide remote access to the database on a record-by-record or field-by-field  
20 basis across the network 12.

21           Further, because the database is configured as a set of files, the  
22 file system 37 may be used to maintain and access the files on the different  
23 media. As a result, the database manager 36 need not be specially  
24 configured to maintain and access the database files 38A, 38B, 38C, and 38D  
25 located on the separate media. Specifically, as will be understood by one  
26 having ordinary skill in the art, file systems are conventionally configured to  
27 access files that are stored on different media. For example, when  
28 configuring a processor and installing a file system, an operator is typically  
29 prompted to define one or more accessible memory drives and to provide a  
30 description of the characteristics of each drive including, for example, whether  
31 the drive is volatile, non-volatile or flash memory. Then, when accessing a  
32 file, the file system uses the pathname of the file to determine which of the

1 memory drives to access and may further use the description of the drive to  
2 determine how to access the drive.

3 In addition, a portion of the VRAM 30 is reserved for usage as a  
4 database cache 35 to which static data stored in the flash memory 32 may be  
5 temporarily copied for modification, should modification be desired. In  
6 addition, the database manager 36 maintains a directory that indicates  
7 whether any of the static data elements have been copied to the cache 35 so  
8 that a user or application 34 trying to access the static data elements has  
9 access to the most recent version of the data elements. As will be appreciated  
10 by one having ordinary skill in the art, using a temporary storage area  
11 constructed to operate as a cache is known in the art and typically involves  
12 reserving a segment of random access memory for temporary data storage.  
13 Typically the memory space occupied by the cache is large enough so that an  
14 entire block of data may be stored in the cache without fragmenting the data  
15 in the VRAM 30. As will further be appreciated by one having ordinary skill in  
16 the art, alternative data storage techniques may be used in place of a cache.  
17 For example, the static data temporarily moved to the cache may be stored  
18 and processed as a file so that the data is treated as a block of data to be  
19 kept together instead of as separate pieces of data.

20 Referring now to FIGs. 2 and 5, a method 50 for creating the  
21 database 38 and installing the database 38 in the application node 18 begins  
22 when a system operator determines and defines the static and dynamic data  
23 fields to be included in the database 38 by creating an initial database file  
24 using the database generation software 43 (step 52). Next, the operator  
25 enters the static data into the fields of the initial database file that are  
26 designated for static data (step 54). After creating the initial database file that  
27 defines the data fields to be included in the database 38 and that includes the  
28 entered, static data, the operator supplies the initial database file to the  
29 database generation software 43 (step 56) and invokes a conversion feature  
30 associated with the database generation software 43 (step 58). Invoking the  
31 conversion feature causes the database generation software 43 to create the  
32 two database files 38A and 38B, formatted as table files, based on the

1 information supplied by the operator. As described above, the first table file  
2 38A contains the catalog 49 that defines the structure of the static fields of the  
3 database 38 and further contains the static data. The second table file 38B  
4 contains the catalog 49 that defines the structure of the dynamic data.

5 As will be appreciated by one having ordinary skill in the art,  
6 database generation software applications that enable the creation of  
7 databases, such as Microsoft Access, are well known in the art and are thus  
8 not described further herein. In addition, dialog software may be implemented  
9 on top of the Microsoft Access program to simplify the data entry process.  
10 Specifically, the dialog software may cause the processor 42 to present a  
11 display containing prompts that inform the user as to the type of data to be  
12 entered into each data field.

13 After the table files 38A and 38B have been created, the user  
14 causes the table files 38A and 38B to be downloaded via the network 12 to  
15 the application node 18 using the well known file transfer protocol (step 59).  
16 At the application node 18, the table files 38A and 38B are stored in the flash  
17 data memory 32B (step 60). With reference now made to FIG. 6, the  
18 application 34 uses the database manager 36 when required to access the  
19 table files 38A and 38B (step 62). As a consequence of this access, the  
20 database manager scans the table catalogs 49 (step 64) and creates cache  
21 entries for static tables (step 66) and dynamic tables (step 68) as well as  
22 dynamic record files 38C and 38D in NVRAM 31 (step 70). Other applications  
23 are free to access these database structures without additional penalties.

24 In addition, dynamic data received from any of the applications  
25 34 or the user is stored in either the NVRAM 31 or the VRAM 30 cache,  
26 depending on how the data is categorized in the catalog 49 (step 66).  
27 Memory space is not reserved in the VRAM 30 and the NVRAM 31 for the  
28 dynamic data elements until the database table files 38A and 38B are  
29 accessed by the application and is then shared by subsequent applications,  
30 thereby ensuring efficient usage of the available VRAM 30 and NVRAM 31.  
31 Further, due to the usage of the file system, the partitioned manner in which  
32 the data elements are stored is transparent to the user and the application

1 programs 34 such that, when extracting data from the database 38 neither the  
2 user nor the application program 34 need know whether the desired data  
3 element is stored in flash 32 or RAM 30.

4 This may be accomplished, for instance, through an application  
5 program 34 creating a functional module or object, such as a CAppDatabase  
6 object, that manages database access. The application 34 may request the  
7 CAppDatabase object to open a table file, which may comprise a pair of files  
8 38A and 38B that have ".tbl" and ".dyn" extensions, respectively, and that  
9 contain a static and a dynamic data portion, respectively. In response, the  
10 object will provide the table name, which is then cached for access to the  
11 associated table files. All static data elements within the ".tbl" file 38A will use  
12 the file named <table\_name>.tbl for a final file update. Dynamic record data  
13 associated with file 38B is stored in a file <table\_name.rec> and contains all  
14 the fields defined in the "a.dyn" 38B catalog.

15 Referring now to FIGs. 4 and 6, the database manager 36  
16 generally maintains the database files 38A, 38B, 38C and 38D according to  
17 the order in which the data fields 48 are listed in their respective catalogs 49.  
18 However, Boolean data elements are treated differently. Specifically, the data  
19 fields 48 in the database are typically defined to be at least a byte long. In  
20 contrast, only a single bit is required to represent a Boolean element, with the  
21 result that a byte-sized space would have eight times the required space. As  
22 a result, storing each Boolean element in a separate byte sized data field 48  
23 represents an inefficient usage of memory. To ensure efficient memory  
24 usage, a novel method 71 may be performed using the conversion feature of  
25 the database generation software application 43 that converts the entered  
26 database structure and entered, static data into the set of database files 38A  
27 and 38B. Specifically, when the initial database file containing the database  
28 structure and the static data is supplied to the database generation software  
29 application (step 72) (see also step 58 of FIG. 5), the conversion feature  
30 identifies any data fields 48 that are designated to contain Boolean data  
31 elements (step 74).

32 Once these data fields 48 have been identified, a data field for

collectively storing the Boolean elements is defined (step 76). In addition, the data fields originally designated to separately store the Boolean elements are eliminated (step 78). Next, the database generation software application 43 generates the database table files 38A and 38B each having a catalog 49 and, depending on the nature of the Boolean data elements, each catalog may define a data field in which the Boolean elements are collectively stored (step 79). Of course, if more than eight Boolean elements are associated with a single data record, then one or more additional bytes are used to store the Boolean elements. In addition, the catalogs 49 reflect the rearranged order of the Boolean elements and the location of each Boolean element is indicated in the catalog file using the name of the data field originally designated to store the Boolean element. Thus, when one of the Boolean elements is accessed by a user operating the workstation 20, the database manager 36 may use the name of the original data field to determine where the Boolean element is stored in the database.

Advantageously, the method 71 thus provides increased memory efficiency by more efficiently using the byte-long data fields 48 that were originally designated to contain a single bit sized Boolean element.

Referring now to FIGs. 2 and 8, the flash memory 32 is arranged in a set of units with each unit capable of holding a fixed amount of data referred to as a page. As is conventional for flash memory, a page of data elements is the smallest amount of data that may be stored in each unit of flash memory at any given time. When modifying the data stored in a unit of flash memory, the unit of flash memory must first be erased before the new data can be stored. Thus, an entire page of flash memory must be erased and written when modifying only a single data element within the page. To avoid having to rewrite an entire page of data into the flash memory 32 each time a single static data element is modified, the present invention provides a method for efficiently modifying the flash memory 32 using the cache memory 39 that may be implemented using software that comprises a subsystem of the database manager 36.

The method 80 begins when a user operating the database

1 interface program 44 attempts to modify a static data element (step 82)  
2 contained in the database 38. Of course, the method 80 may instead begin in  
3 other manners, for example in response to an application 34 attempting to  
4 modify data contained in the database 38. Regardless of the particular  
5 starting step, the database manager 36 subsequently accesses a directory  
6 stored in the RAM 30 that contains information indicating whether the data  
7 element being modified belongs to a static field or a dynamic field (step 84). If  
8 the data element being modified is static, then the data element is modified as  
9 it appears in the cache 35 (step 86) and the static table file 38A field is  
10 overwritten (steps 90, 91). The file system 37 is used to make updates to  
11 static table data; consequently, it will load the flash cache 39 with the  
12 appropriate flash page containing the field to be updated. The new field value  
13 will be written into the flash cache and the cache will be flushed when  
14 finished. If instead the data element is a dynamic field, then the database  
15 manager 36 modifies the data element in the cache 35 in a manner specified  
16 by the user (step 88) and the contiguous memory field in 38C or 38D is  
17 updated without using the file system.

18 For illustrative purposes, the method 80 for modifying the static  
19 data elements is described as being performed at the request of a user  
20 operating the database interface program. However, as described above, any  
21 of the software applications 34 stored in the application node 18 may be  
22 adapted to modify the static element data. Moreover, more than one of the  
23 applications 34, and/or the user may be attempting to modify static data  
24 elements at the same time. In one embodiment, the cache 35 is located in a  
25 portion of the RAM 30 that comprises a shared memory space, i.e., a memory  
26 space accessible to multiple applications. Further, the database manager 36  
27 causes all of the applications to access the same cache 35 when editing the  
28 database 38 so that each application 34 need not create a separate cache  
29 and memory space is conserved.

30 In still additional embodiments of the present invention, the  
31 invention may further comprise connection to an external communications  
32 network for inputting and outputting data. The schematic of FIG. 9 is useful in

1 describing such an embodiment. It will be appreciated that the schematic of  
2 FIG. 9 is consistent in most respects with that discussed above in reference to  
3 FIG. 1. FIG. 9 further illustrates, however, an external communications  
4 network 100 connected to the workstation 20 and to the programmable  
5 controller 16. Through connection to the external network 100, data may be  
6 input or output to control the controller system 10 from a remote site such as  
7 computer 102. It will be appreciated that a wide variety of communications  
8 networks 100 may be comprised within practice of the invention, with  
9 preferred examples comprising the internet, world wide web, telephone  
10 network, proprietary data networks, satellite based networks, and the like.  
11 Further, embodiments of the control system of the invention preferably  
12 support Internet protocol communications for widespread interoperability with  
13 external communications networks such as the Internet.

14 It will also be appreciated that through the external network 100  
15 a variety of devices may be interfaced with the control system 10, with the  
16 computer 102 shown only for illustration. Through any of these devices,  
17 remote control of the control system 10, including but not limited to remote  
18 communication with the database 38, database manger 36, and software  
19 applications 34, may be achieved. Also, a connection to the external network  
20 has been illustrated through work station 20 and through programmable  
21 controller 16 for illustration only, those knowledgeable in the art will appreciate  
22 that connection may occur at any practical point on the communications  
23 network 12, with examples comprising the devices shown as connected  
24 thereto. Further, through an external communications network 100, any of the  
25 various components 14-28 may be remotely located.

26 From the foregoing description, it should be understood that a  
27 database manager and a database for a control system have been described,  
28 having many desirable attributes and advantages. In particular, the database  
29 is configured as a set of files that are stored in different memory devices.  
30 Specifically, the database includes a file containing static data that is stored in  
31 a static memory device and a file containing dynamic data that is stored in a  
32 dynamic memory device. As a result, the overall amount of data stored in

1 dynamic memory is reduced and, thus, dynamic memory is conserved. The  
2 database manager uses a file system to open the database files and access  
3 the data stored therein. Because the database is configured as a set of files,  
4 an application located at a remote network device may access the database  
5 using the well-known file transfer protocol. In addition, a plurality of  
6 applications have access to a cache used to temporarily store static data that  
7 is being modified thereby eliminating the need to create a separate cache for  
8 each application and reducing fragmentation of the dynamic memory.  
9 Further, Boolean elements stored in the database are collected and stored in  
10 a group thereby conserving additional memory.

11 While various embodiments of the present invention have been  
12 shown and described, it should be understood that other modifications,  
13 substitutions and alternatives are apparent to one of ordinary skill in the art.  
14 For example, as described herein, the database and database manager are  
15 disposed in the application node. However, the database and database  
16 manager may instead be disposed in any of the network control devices  
17 provided that the control device includes a processor, static memory device  
18 and a dynamic memory device. Further, although described in the context of  
19 a building control system, the database and database manager may be  
20 implemented in any type of system requiring a data storage system.

21 Various features of the invention are set forth in the appended  
22 claims.